OOP Exercises 2

JavaScript

In the following task, you have the opportunity to write your own JavaScript classes of string and arrays. Yes, we know, that they’re already exiting, but implementing them offers you a great learning opportunity. Not just practicing the OOP approach in JS, but also get a deeper understanding of how arrays and strings work.

## Array implementation

Create a class, called Array, which’s constructor returns an empty object. The object has a length property, which stores how many items are stored in the array. The object stores the individual items as a key-value pair, where the key is identical to the index of the item in the array.

The class has multiple built-in methods, like:

* Array.push: add a new item to the end of the object,
* Array.pop: delete the last item of the array,
* Array.valueOf: this method takes a number as a parameter, and returns the associated value. If there is no value associated with that index, the method returns an error message.

Example:

|  |
| --- |
| ***let*** *numbers =* ***new*** *Array() // creates a new array object numbers.push(12); // add new item to the array numbers.push(45); numbers.push(3); numbers.push(21); numbers.push(89);  /\* The object contains all the items above numbers = {  length: 5,  0: 12,  1: 45,  2: 3,  3: 21,  4: 89 } \*/  numbers.pop(); // remove the last element of the array  /\* The object contains all the items above numbers = {  length: 4,  0: 12,  1: 45,  2: 3,  3: 21 } \*/  numbers.valueOf(3) // returns 21* |

If you are keen on extra challenges (💪💪) you can implement an Array.splice method, which takes two parameters (Numbers), a start and end index, and returns a new instance of the Array class containing all the items between those indexes.

## String

In this exercise, you will implement the functioning of strings in JavaScript. Create a class, called String. Its constructor takes one parameter, a string, and store it as an array of characters, e.g.:

|  |
| --- |
| */\* input: "Hello World!" the String instance should store the characters as the followings: {  0: "H",  1: "e",  2: "l",  3: "l",  4: "o",  5: " ",  6: "W",  7: "o",  8: "r",  9: "l",  10: "d",  11: "!" } \*/* |

The object has a length property, which stores the length of the string. Besides that, it also has some built-in methods, like:

* String.concat(<String>) -> takes one parameter, another string and adds to the original one
* String.charAt(<number>) -> takes a Number as parameter and returns the associated value.
* String.indexOf(<character>) -> takes a character as parameter and returns the first index of the character.

If you like extra challenges (💪💪) you write an extra method, called String.includes(), which takes a string as a parameter, and returns true, if the original string includes that string, and false otherwise.

## Expand Array class 💪💪

If you are looking for further challenges, you can implement the Array.forEach() method, which takes one parameter, a callback function, and iterates through the array, and calls the function on every item, and as the last step, it stores the modified value in the original array.